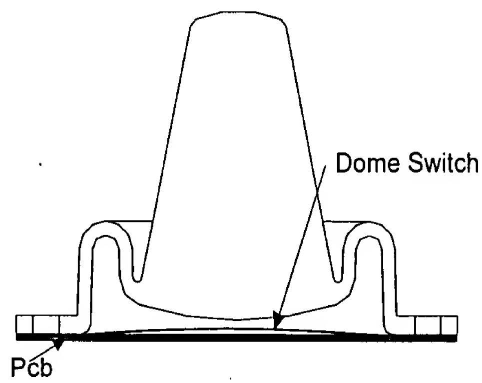


Fig 1: Not in "Presence"

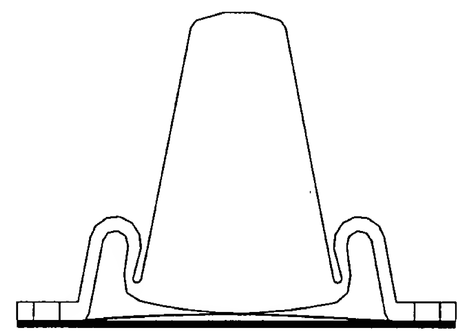


Fig 2: In "Presence"

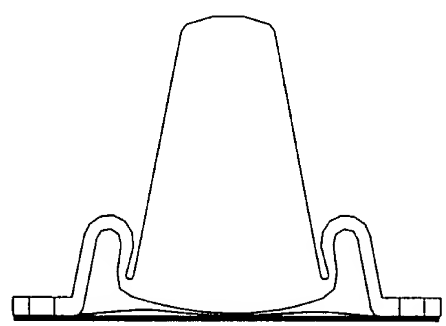


Fig 3: Clicking (Dome Switch collapsed)

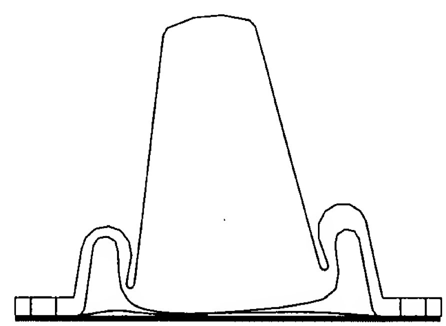
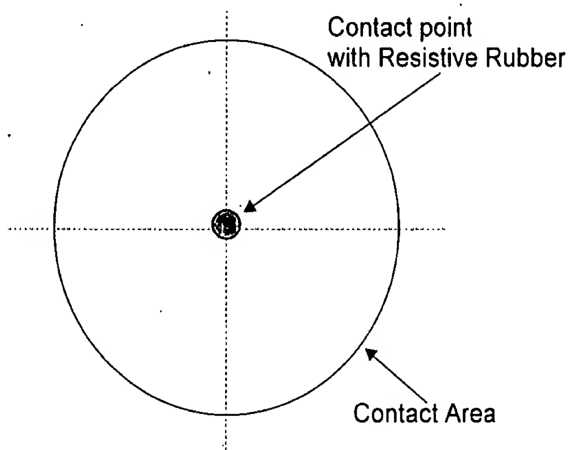


Fig 4: Clicking "On the Fly"



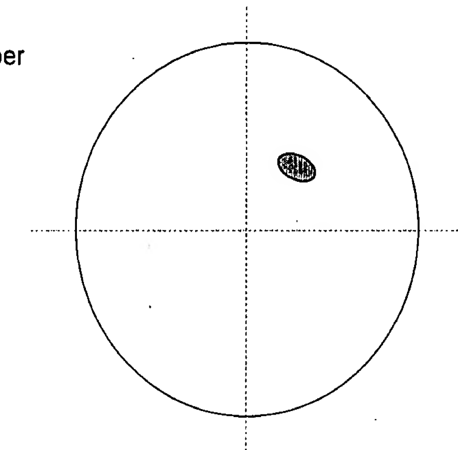


Fig 5: Typical "Ninja landing" contact   Fig 6: Typical "hit the decks running" contact

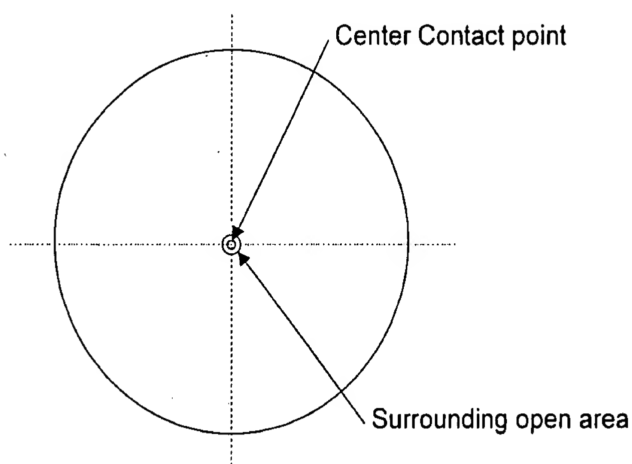


Fig 7: "Hardware Help" for auto-calibration

09523407 .031000

eri.

Include/File structure for
VaraPoint Firmware

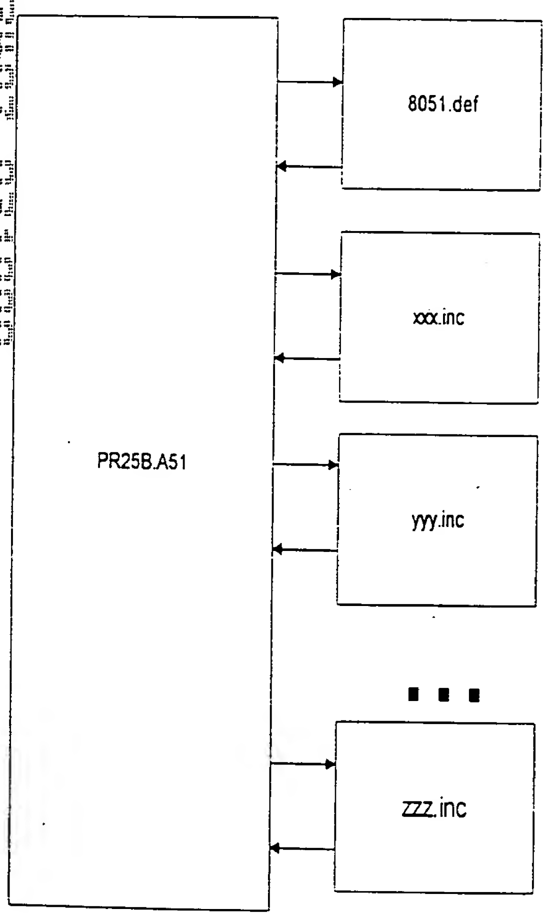


Fig. 8

tions a separate switch is provided, which may be scanned in other code, or by some other processor.

c) SNAPTOGRID

This switch determines whether code will be generated to cause motion near a multiple of 90 degrees to "snap" to those directions. This switch is normally turned on when the application involves GUI (Graphical User Interface) menu navigation, etc. It would be less desirable if the primary application involved sketching and drawing.

d) NAVIGATEMENU

This switch determines whether code will be generated to cause motion near a multiple of 45 degrees will "snap" to those directions. It is much like the SNAPTOGRID switch, but allows "snapping" to the true axes and the 45-degree diagonals. As with SNAPTOGRID, the intended application would dictate whether this switch should be set.

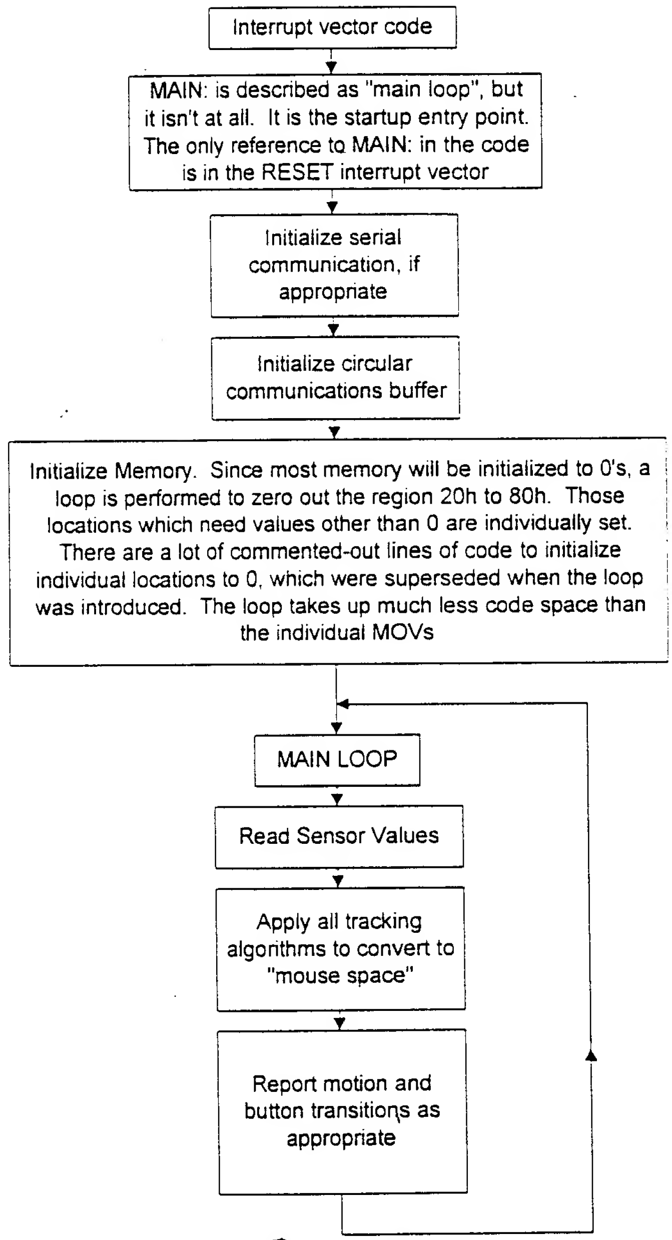


Fig. 9

e) AUTOCENTER

This switch determines whether code will be generated to cause the system to re-calibrate itself for "centering", the position at which no motion is generated. In any joystick or joystick-like pointing device, any of a number of situations can cause the "null" position to be other than where the user expects it. This can cause a perception of "skating in the wind", or "drift". This switch would normally be turned on, except perhaps for some unusual applications in which the autocentering behavior conflicts with other design goals. With this switch turned on, the system automatically re-calibrates itself based on a rolling average of contact points, with automatic compensation for outlying values.

icrocontroller reads a minimum deflection at a slightly larger than minimum ical deflection on a nominal VaraPoint (this must be the case for a nominal setup so we can always attain a minimum deflection taking variation of component values account). As deflection is increased, the ge time is decreased, since the direction's tance to Vcc is decreased. If the charge is longer than the 1.397 msec sampling low, this is read as no deflection at all.

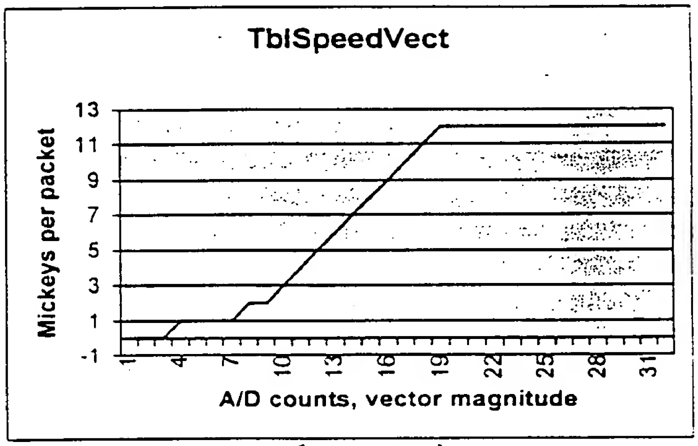


Fig. 10A

e-based parameters

TblSpeedVect and TblDelayVect tables together to implement the main part of Point tracking. These tables will normally djusted together to account for differences icroprocessor speed and A/D circuits (such as different reference voltages, resistor capacitor values, etc.). Together, these tables implement a "Three Plateaus" approach to tracking: *Fine Control*, *Naviga-* and *Blitz*. A general strategy for adjusting these tables would be to compare a trial ementation with a reference design Vara-t, and to try to achieve a similar level of rol. Fine-tuning would best be accomplished with user-level tests on small subject ps using VaraTouch's Pointer Evaluation ware, its Fitt's Law suite in particular.

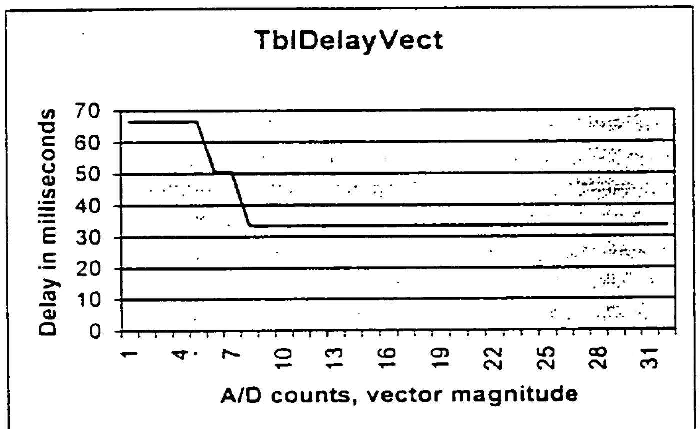


Fig. 10B

TblSpeedVect

This is the primary tracking table, working in conjunction with TblDelayVect. It contains 32 entries, for each of 32 possible input counts (coming from the A/D circuit). For each input count, it gives the number of output counts ("*Mickeys*") that the firmware should report. These *Mickeys* correspond to the counts normally made by optical encoders in a typical 300-dpi mouse. Thus, when the A/D measures a force corresponding to NNN counts from the A/D, the system will report TblSpeedVect(NNN) Mickeys on

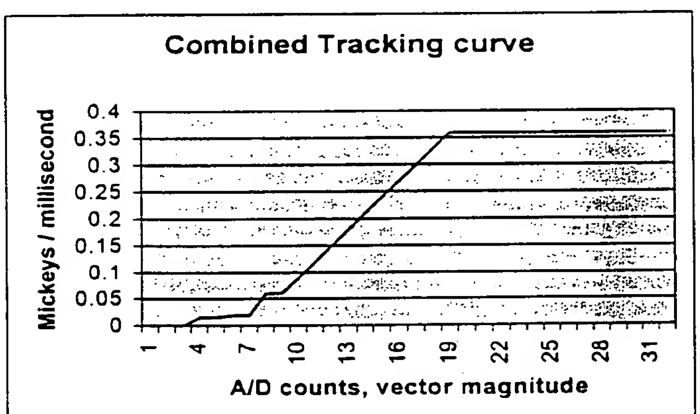


Fig. 10C

eport out, which will occur every TblDelayVect(NNN) milliseconds.

blDelayVect

his table works in conjunction with TblSpeedVect to condition the apparent rate of otion reported by the firmware. The TblSpeedVect/TblDelayVect system is an al-ernative to "Fractional Mickey" tracking, which would have to be maintained if the otion would correspond less than one Mickey usually 1/300 of an inch) er packet of data sent out. y delaying the time be-veen packets, the effec-ve rate of motion is kept ppropriately low when the tended motion is slow.

blSlowVect

his table manages an im-ortant aspect of Vara-oint tracking – an alterna-ve tracking during decel-ration to manage the overshoot" which so of-en otherwise characterizes oystick pointing. During acking, the firmware al-ays remembers the last orce vector magnitude, nd continually compares to the current force vec-or magnitude. When the agnitude is decreasing, en the user is attempting slow down. The differ-nce between last and cur-ent magnitude will be ositive during decelera-on, and is used as an index into the TblSlowVect to calculate an adjustment to the peed to help slow down motion faster and minimize overshoot. The value in the ble is mutiplied by previous (larger) magnitude, and that number is subtracted from e current magnitude.

Tracking Algorithm Description

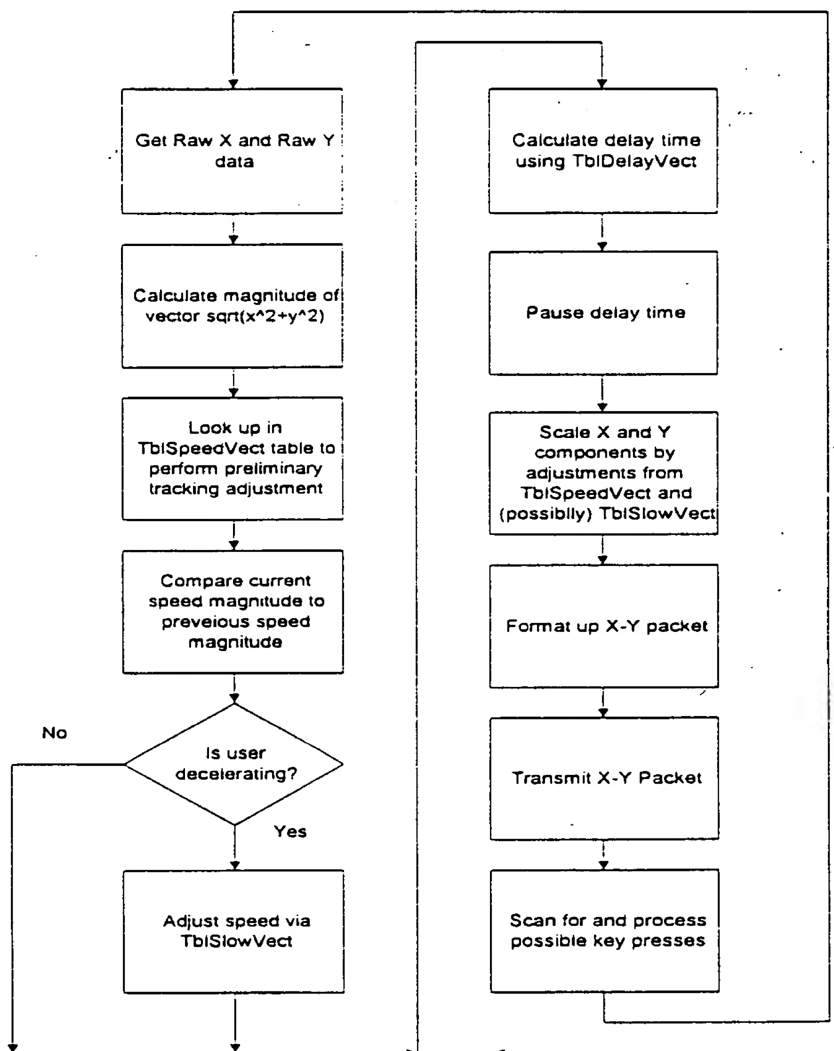


Fig. 11

m Hardware Requirements